Top Interview Questions with Java Solutions

1. Two Sum - Find two numbers that add up to a target

```
public int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < nums.length; i++) {
        int complement = target - nums[i];
        if (map.containsKey(complement)) {
            return new int[] { map.get(complement), i };
        }
        map.put(nums[i], i);
    }
    return new int[] {};
}
```

2. Longest Substring Without Repeating Characters

```
public int lengthOfLongestSubstring(String s) {
    Set<Character> set = new HashSet<>();
    int left = 0, maxLen = 0;
    for (int right = 0; right < s.length(); right++) {
        while (set.contains(s.charAt(right))) {
            set.remove(s.charAt(left++));
        }
        set.add(s.charAt(right));
        maxLen = Math.max(maxLen, right - left + 1);
    }
    return maxLen;
}
```

3. Kadane's Algorithm - Maximum Subarray Sum

```
public int maxSubArray(int[] nums) {
    int maxSum = nums[0], currSum = nums[0];
    for (int i = 1; i < nums.length; i++) {
        currSum = Math.max(nums[i], currSum + nums[i]);
        maxSum = Math.max(maxSum, currSum);
    }
    return maxSum;
}
```

4. Merge Intervals

```
public int[][] merge(int[][] intervals) {
    Arrays.sort(intervals, (a, b) -> a[0] - b[0]);
    List<int[]> merged = new ArrayList<>();
    for (int[] interval : intervals) {
```

```
        if (merged.isEmpty() || merged.get(merged.size() - 1)[1] < interval[0]) {
            merged.add(interval);
        } else {
                merged.get(merged.size() - 1)[1] = Math.max(merged.get(merged.size() -
1)[1], interval[1]);
        }
    }
    return merged.toArray(new int[merged.size()][]);
}
```

5. Valid Parentheses

```
public boolean isValid(String s) {
    Stack<Character> stack = new Stack<>();
    for (char c : s.toCharArray()) {
        if (c == '(') stack.push(')');
        else if (c == '{') stack.push('}');
        else if (c == '[') stack.push(']');
        else if (stack.isEmpty() || stack.pop() != c) return false;
    }
    return stack.isEmpty();
}
```